**AMENDMENTS TO THE CLAIMS**

The listing of claims will replace all prior versions, and listings, of claims in the application:

**Listing of Claims:**

1. (Currently Amended) A computer-implemented method to load objects in a heterogeneous multiprocessor computer system, said method comprising:

analyzing a source program for one or more program characteristics, the program characteristics selected from the group consisting of data locality, computational intensity, and data parallelism;

in response to the analyzing, compiling the source program into two object files, a first object file corresponding to a first instruction set and a second object file corresponding to a second instruction set;

in response to the compiling, storing the program characteristics in each of the object files;

receiving a request to execute a software task corresponding to the source program;

~~identifying~~ selecting a processor from a plurality of dislike processors, wherein a first processor corresponds to the first instruction set and a second processor corresponds to the second instruction set, to execute [[a]] the software task, the ~~identification based upon characteristics of the software task and computing resource availability~~ selecting comprising comparing one or more characteristics of the software task with the program characteristics stored in the first object file and the second object file;

in response to selecting the first processor:

loading ~~software code corresponding to the identified processor~~ the first object file into a shared memory, wherein the shared memory is shared by

[[a]] the plurality of dislike processors ~~that includes the identified processor~~; and

executing the loaded [[code]] first object file by the ~~identified~~ first processor[[.]]; and

in response to selecting the second processor:

loading the second object file into the shared memory; and

executing the loaded second object file by the second processor.

2. (Canceled)

3. (Canceled)

4. (Canceled)

5. (Currently Amended) The method as described in claim [[3]] 1 wherein ~~identifying~~ selecting the processor further comprises:

retrieving the program characteristics;

retrieving current system characteristics, wherein the current system characteristics includes processor load characteristics for the plurality of dislike processors; and

combining the program characteristics and the current system characteristics to determine which of the dislike processors to assign the software task.

6. (Original) The method as described in claim 5 wherein at least one of the current system characteristics is selected from the group consisting of processor availability for each of the dislike processors, and a data size of data being processed by the software task.

7. (Currently Amended) The method as described in claim 1 wherein executing the first object file by the first processor further ~~comprising~~ comprises:

determining that the ~~identified~~ first processor has a scheduler that schedules tasks for the first processor; and

scheduling the ~~software code~~ first object file to execute on the ~~identified~~ first processor, the scheduling including:

writing a software code identifier corresponding to the ~~software code~~ first object file to a run queue corresponding to the ~~identified~~ first processor.

8. (Currently Amended) The method as described in claim 1 wherein executing the second object file by the second processor further ~~comprising~~ comprises:

signaling the ~~identified~~ second processor; and

reading, by the ~~identified~~ second processor, the ~~software code~~ second object file from the shared memory into a local memory corresponding to the ~~identified~~ second processor~~; and~~

~~executing the software code by the identified processor~~.

9. (Currently Amended) The method as described in claim 8 further comprising:

writing an instruction block in the shared memory, the instruction block including [[the]] an address of the loaded ~~software code~~ second object file and [[the]] an address of an input buffer; and

reading the ~~software code~~ loaded second object file and the input buffer from the locations identified in the instruction block to the ~~identified~~ second processor 's local memory.

10. (Currently Amended) The method as described in claim 9 further comprising:

signaling the ~~identified~~ second processor from one of the other processors, the signaling including:

writing the address of the instruction block to a mailbox that corresponds to the ~~identified~~ second processor; and

reading, by the ~~identified~~ second processor, the instruction block in response to the signal.

11. (Currently Amended) An information handling system comprising:

a plurality of heterogeneous processors;

a common memory shared by the plurality of heterogeneous processors;

a first processor selected from the plurality of processors that sends a request to a second processor, the second processor also being selected from the plurality of processors, wherein the first processor corresponds to a first instruction set and the second processor corresponds to a second instruction set;

a local memory corresponding to the second processor;

a Direct Memory Access (DMA) controller associated with the second processor, the DMA controller transferring data between the common memory and the second processor's local memory; and

a loading tool to load software code to execute on one of the processors, the loading tool including software effective to:

analyze a source program for one or more program characteristics, the program characteristics selected from the group consisting of data locality, computational intensity, and data parallelism;

compile the source program into two object files, a first object file corresponding to a first instruction set and a second object file corresponding to a second instruction set;

store the program characteristics in each of the object files;

receive a request to execute a software task corresponding to the source program;

[[identify]] select one of the processors to execute [[a]] the software task, ~~the identification based upon characteristics of the software task and computing resource availability~~ by comparing one or more characteristics of the software task with the program characteristics stored in the first object file and the second object file;

in response to selecting the first processor:

[[loading]] load the ~~software code corresponding to the identified processor~~ first object file into the common memory; and

~~executing~~ execute the loaded [[code]] first object file by the ~~identified~~ first processor[[.]]; and

in response to selecting the second processor:

load the second object file into the common memory; and

execute the loaded second object file by the second processor.

12. (Canceled)

13. (Canceled)

14. (Canceled)

15. (Currently Amended) The information handling system as described in claim [[13]] 11 wherein ~~identification~~ selection of the processor further comprises software effective to:

retrieve the program characteristics;

retrieve current system characteristics, wherein the current system characteristics includes processor load characteristics for the plurality of heterogeneous processors; and

combine the program characteristics and the current system characteristics to determine which of the heterogeneous processors to assign the software task.

16. (Original) The information handling system as described in claim 15 wherein at least one of the current system characteristics is selected from the group consisting of processor availability for each of the heterogeneous processors, and a data size of data being processed by the software task.

17. (Currently Amended) The information handling system as described in claim 11 further comprising software effective to:

determine that the ~~identified~~ first processor has a scheduler that schedules tasks for the first processor; and

schedule the ~~software code~~ first object file to execute on the ~~identified~~ first processor, the ~~schedule~~ scheduling including software effective to:

write a software code identifier corresponding to the ~~software code~~ first object file to a run queue corresponding to the ~~identified~~ first processor.

18. (Currently Amended) The information handling system as described in claim 11 further comprising software effective to:

signal the ~~identified~~ second processor; and

read, by the ~~identified~~ second processor, the ~~software code~~ second object file from the common memory into a local memory corresponding to the ~~identified~~ second processor~~; and~~

~~execute the software code by the identified processor~~.

19. (Currently Amended) The information handling system as described in claim 18 further comprising software effective to:

write an instruction block in the common memory, the instruction block including [[the]] an address of the loaded ~~software code~~ second object file and [[the]] an address of an input buffer; and

read the ~~software code~~ loaded second object file and the input buffer from the locations identified in the instruction block to the ~~identified~~ second processor 's local memory.

20. (Currently Amended) The information handling system as described in claim 19 further comprising software effective to:

signal the ~~identified~~ second processor from one of the other processors, the signal including software effective to:

write the address of the instruction block to a mailbox that corresponds to the ~~identified~~ second processor; and

read, by the ~~identified~~ second processor, the instruction block in response to the signal.

21. (Currently Amended) A computer program product stored in a computer operable media to load objects in a heterogeneous multiprocessor computer system, said computer program product comprising instructions that, when executed by an information handling system, cause the information handling system to perform steps comprising:

analyzing a source program for one or more program characteristics, the program characteristics selected from the group consisting of data locality, computational intensity, and data parallelism;

in response to the analyzing, compiling the source program into two object files, a first object file corresponding to a first instruction set and a second object file corresponding to a second instruction set;

in response to the compiling, storing the program characteristics in each of the object files;

receiving a request to execute a software task corresponding to the source program;

~~identifying~~ selecting a processor to execute a software task, from a plurality of dislike processors, wherein a first processor corresponds to the first instruction set and a second processor corresponds to the second instruction set, the ~~identification based upon characteristics of the software task and computing resource availability~~ selecting comprising comparing one or more characteristics of the software task with the program characteristics stored in the first object file and the second object file;

in response to selecting the first processor:

loading ~~software code corresponding to the identified processor~~ the first object file into a shared memory, wherein the shared memory is shared by [[a]] the plurality of dislike processors ~~that includes the identified processor~~; and

executing the loaded [[code]] first object file by the identified processor[[.]]; and

in response to selecting the second processor:

loading the second object file into the shared memory; and

executing the loaded second object file by the second processor.

22. (Canceled)

23. (Canceled)

24. (Canceled)

25. (Currently Amended) The computer program product as described in claim [[23]] 21 wherein the ~~identifying~~ selecting the processor further comprises:

retrieving the program characteristics;

retrieving current system characteristics, wherein the current system characteristics includes processor load characteristics for the plurality of dislike processors; and

combining the program characteristics and the current system characteristics to determine which of the dislike processors to assign the software task.

26. (Original) The computer program product as described in claim 25 wherein at least one of the current system characteristics is selected from the group consisting of processor availability for each of the dislike processors, and a data size of data being processed by the software task.

27. (Currently Amended) The computer program product as described in claim 21 wherein ~~the steps further comprise~~ executing the first object file by the first processor further comprises:

determining that the ~~identified~~ first processor has a scheduler to schedule tasks for the first processor; and

scheduling the ~~software code~~ first object file to execute on the ~~identified~~ first processor, the scheduling including:

writing a software code identifier corresponding to the ~~software code~~ first object file to a run queue corresponding to the ~~identified~~ first processor.

28. (Currently Amended) The computer program product as described in claim 21 wherein ~~the steps further comprise~~ executing the second object file by the second processor further comprises:

signaling the ~~identified~~ second processor; and

reading, by the ~~identified~~ second processor, the ~~software code~~ second object file from the shared memory into a local memory corresponding to the ~~identified~~ second processor~~; and~~

~~executing the software code by the identified processor~~.

29. (Currently Amended) The computer program product as described in claim 28 wherein the steps further comprise:

writing an instruction block in the shared memory, the instruction block including [[the]] an address of the loaded ~~software code~~ second object file and [[the]] an address of an input buffer; and

reading the ~~software code~~ loaded second object file and the input buffer from the locations identified in the instruction block to the ~~identified~~ second processor 's local memory.

30. (Currently Amended) The computer program product as described in claim 29 wherein the steps further comprise:

signaling the ~~identified~~ second processor from one of the other processors, the signaling including:

writing the address of the instruction block to a mailbox that corresponds to the ~~identified~~ second processor; and

reading, by the ~~identified~~ second processor, the instruction block in response to the signal.